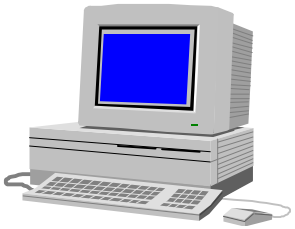
ASIST Level 2 Training for EO-1



ASIST RDL Development

Using an Editor

Read the User's Guide

I&T and Flight Operations

Building the Database

Syntax Verification

Limit Definitions

Database Control

Display Page Development

Notes on Using STOL

Directory Structure

ASIST RDL Development

- RDL is an acronym for Record Definition Language
- The RDL structure is similar to C data definitions with RDL unique keywords
- A record is the name given to a definition structure in the ASIST command and telemetry database (i.e. individual mnemonic definitions). Records are defined using the keyword CLASS for commands, and PACKET for telemetry.
- Each database item must have a unique primary mnemonic. Sub-mnemonic keywords for commands may be re-used . A common sub-mnemonic field example is: ON or OFF.
- APID (Application Identification) is the common name given to a text file created to define a series of records. Formal naming requirements for these files for EO-1 are: eo1cmdxxx.rdl, eo1tlmxxx.rdl, and eo1pstlmxxx.rdl, where xxx is the APID in HEX format.

ASIST RDL Development (cont.)

- APID assignments are controlled by the C&DH FSW.
- ASIST has 4 database areas: system, configured, global, and local.
- **The System area MUST be left alone, these are the files used to create the STOL runtime environment.**
- **The Configured area is read only - files may be copied from configured to a read/write area. This area is reserved to provide configuration control of database files.**
- **The Global and Local areas are common to all workstations in an NFS lab environment.**
- **The PRIVATE area of ASIST is unique to each particular workstation.**
- RDL examples are provided in the global RDL directory as part of the ASIST environment for commands and telemetry.
- The example files are named: eo1tlmex.rdl, eo1cmdex.rdl, and eo1pstlmex.rdl. These files may be copied and used for cut and paste revisions.

Using an Editor

- ASIST is a UNIX based platform, therefore all ASCII characters are represented in both cases. i.e. “A” does not equal “a”.
- All mnemonic names WILL BE in UPPER CASE letters.
- All RDL file names will be in lower case letters.
- All STOL procedure names will be in lower case letters.
- All DISPLAY page names will be in UPPER CASE letters.
- **DO NOT** mix and match cases!
- ASIST comes with an “emacs” editor installed. An EDT emulator for the right numerical keypad is also provided. Database editing can be initiated from the STOL command line or in any X-term window.
- An emacs tutorial is available by starting the emacs editor in an X-term window (emacs anyname), then using the emacs HELP pull down menu to start the tutorial.

Using an Editor (cont.)

- The standard editor tool is entered from the STOL command line. At the STOL prompt type: dbedit <return> to edit RDL files, or prcedit <return> to edit STOL procedures.
- The standard editor uses a LISP interactive menu. Most new users of UNIX may find the X-term emacs approach easier to utilize and understand for creating new files.
- Some warnings on the hazards of using UNIX:

– NEVER use the UNIX delete command ‘rm’ with the asterix.

– NEVER assume files have been copied to a directory. Verify the copy command was successful.

– UNIX only saves 2 copies of a file: the newest and last versions.

– UNIX will allow you to copy over a file or replace a file without saving the older version.

Read the User's Guide

- The ASIST User's guide is an excellent source for reference. The Guide is best used interactively and is available on-line. The ASIST User's guide is just that. All questions will not necessarily be answered
- ASIST is largely a User defined system. Familiarity with UNIX standards and tools (such as grep, awk, ls, and man) make for a better user. Do NOT attempt to learn how to use these tools during I&T activities.
- ASIST operates using a COTS system called SAMMI. SAMMI is a GUI system used to display telemetry mnemonics to the user. SAMMI allows for importation of BMP, GIF or XBM files as static objects. The User's guide only details how to use SAMMI, not how to program with SAMMI.
- The strength of the system is its flexibility.

I&T and Flight Operations

- ASIST is being groomed to be a complete I&T system by the MIDEX program.
- EO-1 is using ASIST as an end to end I&T and FOT system also. Additions to the database for use and clarification of data for Flight Operations is encouraged. A prime example is the keyword TVER. TVER is used for end-to-end command verification. TVER is included in the eo1cmdex.rdl examples and should be included in command mnemonic definitions, but will not necessarily be expanded on by the I&T team. EO-1 I&T will primarily rely on COP-1 and telemetry response for end-to-end command verification.
- Other examples: header information on packet definitions shall be complete enough to determine packet source and downlink time requirements, STOL procedure header information will be sufficient enough to determine hazards, constraints, and warnings by a person unfamiliar with the procedure.

Building the Database

- There are 3 compilers used to build the complete I&T database: Command compiler, Telemetry compiler, and a STOL compiler.
  NOTE: The use of the STOL compiler is addressed in a later section.
- To compile an RDL file into the database, the APID file name must be included in the user_cmd.rdl or user_tlm.rdl file. Most user's will find it easier to select a file name (such as fswlabcmd.rdl) that always remains in user_...rdl instead of continuously updating the user_...rdl files. Edit the file fswlabcmd.rdl to add and delete individual APID packet definitions. This will help prevent over writing of files with the same names or accidentally deleting required file names from the lab's database.
- The RDL compiler builds binary code output for the ASIST system in the area that the user is logged in under. These binary files may not be copied upwards (i.e. local to global, or global to configured).
- ASIST is designed so that the source code will reliably build the same output database (binary code) each time the same source code is used.

Building the Database (cont.)

- Creating the RDL database binary file is a 2 step process comprised of a build followed by a load. The build process creates files named cmd.db or tlm.db and takes ~4 minutes to complete. The load process copies the ‘‘new” binary database file over the “old” binary file and causes the old file to cease to exist (unless saved in another directory). An error in compilation will prevent a *.db file from being created.
- The ASIST STOL environment must be exited and re-started to use the new version of the database. Each individual workstation in an NFS environment must be re-started to use the “new” database.
- The commands for creating a database are: DBCMPCMD for command RDLs (user_cmd.rdl), and DBCMPTLM for telemetry RDLs (user_tlm.rdl). These directives may be typed in at a UNIX prompt or at the STOL command line.
- The commands to load the database are DBLOADCMD and DBLOADTLM.

Syntax Verification

- Each compiler can be run in a syntax mode which does not create a binary output file, but does verify that the syntax required to build a database is valid.
- Error and warning messages are provided to the user after completion of a syntax check or an actual database build.
- Run syntax checks fairly often to find where bad code exists and prevent its re-use.
- A syntax check must be run from a UNIX prompt. Syntax checks may be run against all RDL files, or individual RDL files. The user should be in the ‘rdl’ directory to perform a syntax check. The commands to perform a syntax check are: ‘ocmd eo1cmdxxx.rdl’ for command RDLs, or ‘otlm eo1tlmxxx.rdl’ for telemetry RDLs.
- The syntax checker will not catch every error in the database. A database report should be generated and reviewed for accuracy. This is the best way to catch typographical errors in certain fields.

Limit Definitions

- Limit definitions are imbedded into the RDL file structure. Not ALL telemetry items require limit definitions.
- Limits are structured as follows: RED HIGH, YELLOW HIGH, YELLOW LOW, and RED LOW.
- RED limits are defined as limits which will cause potential damage to the spacecraft on orbit (or during test).
- YELLOW limits are WARNINGS that a potentially hazardous condition may occur.
- A Yellow limit shall be defined so that there is time for corrective action prior to reaching the RED limit condition.
- RED and YELLOW limits are managed ON and OFF by using STOL.
- Limit checking is hierarchical: limit checking must be enabled for an individual ASIST workstation (CHECK ON), limit checking for the packet must be enabled (LIM ON P0xx), and limit checking for individual telemetry points must be enabled (LIM ON SCIERRQ)

Limit Definitions (cont.)

- Limit definitions can be inclusive or exclusive for boundary definitions
- Limit checking generates an event message every time a packet is received by ASIST with an out of limit (OOL) item. A STOL event message is generated for each OOL item's state.
- RED limit violations HALT STOL procedures on receipt. Action must be taken to clear the RED condition prior to continuation of the nominal STOL process (i.e. CHECK OFF, LIM OFF mnemonic, or correct the state which caused the limit violation).
- YELLOW limit violations show as event messages in the STOL environment, but do NOT halt STOL procedure execution.
- Page display fields change to RED or YELLOW for an item, provided that the DDO quality field is enabled for Run Time Annotations (RTA)
- The DDO quality field also shows packet staleness states (telemetry item never received or not received in predetermined time period). The staleness time is defined in the RDL telemetry Packet structure .

Database Control

- Development of RDL files, STOL procedures, and telemetry PAGE displays is done by various engineers in different lab locations. The mechanism to prevent conflicts between labs is the EO-1 Project Database Management Plan [AM149-0048(155)]. Be familiar with the EO-1 database requirements plan prior to performing development work on deliverable items. Be familiar with: mnemonic naming conventions, STOL procedure naming conventions, display page naming conventions, and APID responsibilities.
- Control of database deliverables for individual labs is the responsibility of the lab manager.
- Not establishing a database management function for a lab environment can lead to using up disk space on the workstation.

Display Page Development

- Pages are developed using a graphical interface tool to link database items (Dynamic Display Objects - DDOs) to a display window, commonly called a telemetry page.
- This graphical tool is SAMMI. SAMMI is being used to support the Space Station and Space Shuttle programs for NASA in addition to being used for ASIST.
- The page editor is active when the editor's interactive window is present.
- There are four basic DDO types: dynamic(text), integer, real, and time. Format control of these four types is the responsibility of the page developer.
- Graphic tools are extensive. Some ASIST standard items: panel meters, toggle buttons, bar graphs, X Y plots, and sliding scale meters
- Page editing is most quickly performed by copying in an aggregate style page and changing internal mnemonic definition read keys, page layout and text fields.
- Once a DDO field type is selected, it can be changed to be another type (i.e. change a DYNAMIC (TEXT) to an INTEGER or a TIME field)

Notes On Using STOL

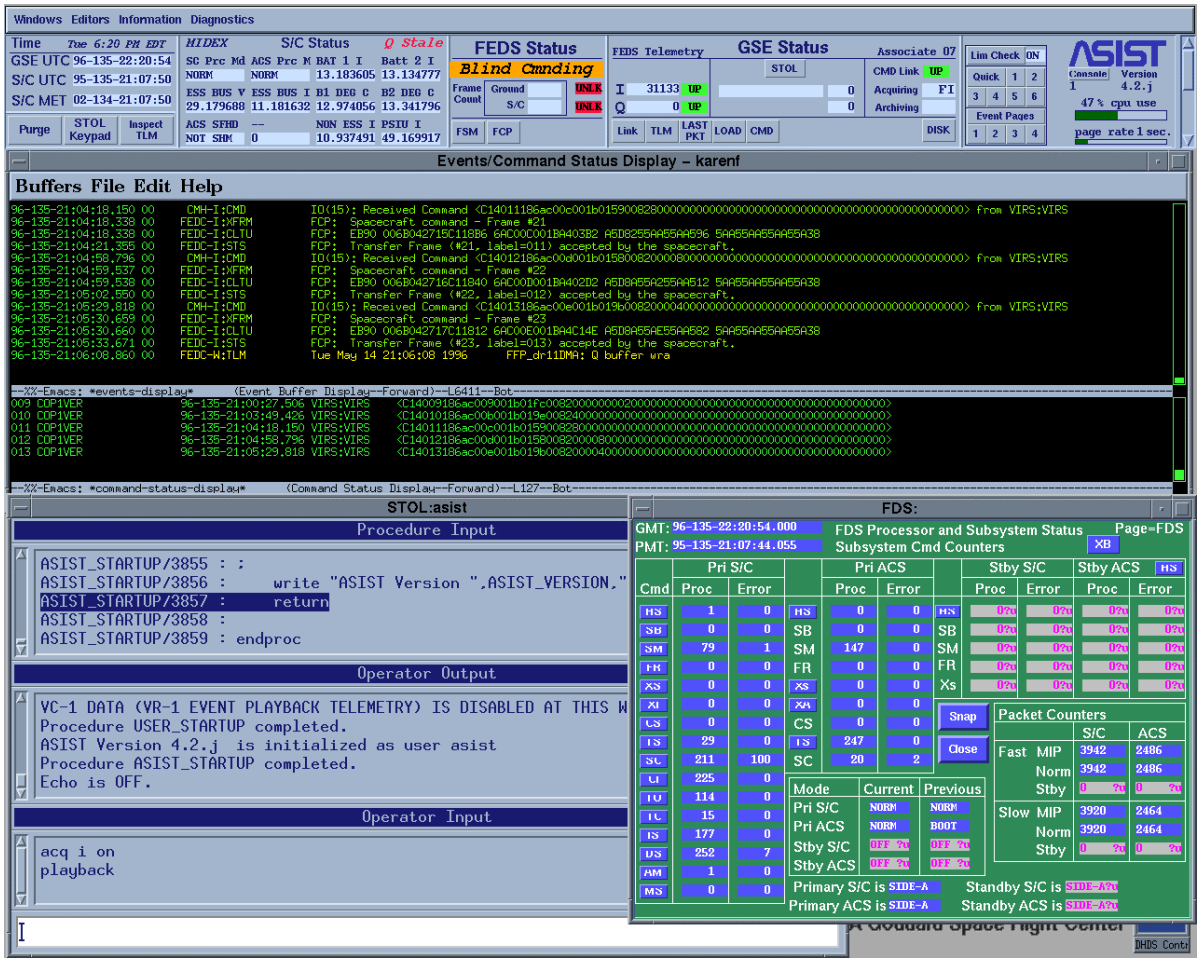
- STOL provides the primary interface between the Test Conductor and the EO-1 Spacecraft.
- STOL is a series of user generated scripts which are developed to perform spacecraft functional verification.
- ASIST foreign directives are defined by execution of STOL scripts. Special commands are currently defined as foreign directives.
- The ASIST runtime environment is defined by execution of STOL scripts. The ASIST user will define functions used in the STOL environment by use of a STOL script named "user_startup.prc". This script is the holding place for items which are defined at system initialization and should be common to all ASIST EO-1 labs.
- STOL procedures must be compiled to create executable files. At the STOL window type "STOL_COMPILER your_proc_name" to compile a procedure. The compiler will provide a summary of errors when completed. The output of the compiler is an executable procedure.

Directory Structure

- ASIST requires that source code files be located in the correct directory. The directory structure for management of ASIST files is listed here:
  - DB - contains binary database files, spacecraft event message file(s), and configuration files for the ASIST, FEDS, and associated GSE
  - IMAGE - holding directory for FSW tables, images, and patches
  - RDL - contains command and telemetry source code
  - PRC - contains STOL procedures
  - SAM - contains SAMMI telemetry page displays
  - **Note that for an NFS system, all directories are common to all machines.**
- On opening an X-term window, the user is placed in the private directory. The private area is unique to each individual workstation and may be used by as desired by the responsible engineer(s).

Topics Not Covered by This Training

- Specific Details on Database Construction
- Technical Details on Telemetry and Command Processing
- Technical Details on Telemetry Page Development
- Directives to Control FEDS and the DHDS
- ASIST Directives for Physical Channel Control & Management
- Local Archives versus DHDS Archives
- Standard Data Format Units (SFDUs), COP-1, and CCSDS formats
- Primary Workstation Command Screening
- Project message files
- Further information is available at **http://rs733.gsfc.nasa.gov/ASIST**


Windows Editors Information Diagnostics
Time Tue 6:20 PM EDT
GSE UTC 96-135-22:20:54
S/C UTC 95-135-21:07:50
S/C MET 02-134-21:07:50
Purge
STOL Keypad
Inspect TLM
HIDEX
S/C Status
Q Stale
FEDS Status
Blind Cmnding
GSE Status
Associate 07
CMD Link UP
Acquiring FI
Archiving
ASIST
Console Version
1 4.2.j
47 % cpu use
page rate 1 sec.
Events/Command Status Display – karenf
Buffers File Edit Help
96-135-21:04:18.150 00 CMH-I:CMD IO(15): Received Command <C14011186ac00c001b0159008280000000000000000000000000000000000000000000000000> from VIRS:VIRS
96-135-21:04:18.338 00 FEDC-I:XFRM FCP: Spacecraft command - Frame #21
96-135-21:04:18.338 00 FEDC-I:CLTU FCP: EB90 006B042715C118B6 6AC00C001BA403B2 A5D8255AA55AA596 5AA55AA55AA55A38
96-135-21:04:21.355 00 FEDC-I:STS FCP: Transfer Frame (#21, label=011) accepted by the spacecraft.
96-135-21:04:58.796 00 CMH-I:CMD IO(15): Received Command <C14012186ac00d001b0158008200008000000000000000000000000000000000000000000000> from VIRS:VIRS
96-135-21:04:59.537 00 FEDC-I:XFRM FCP: Spacecraft command - Frame #22
96-135-21:04:59.538 00 FEDC-I:CLTU FCP: EB90 006B042716C11840 6AC00D001BA402D2 A5D8A55A255AA512 5AA55AA55AA55A38
96-135-21:05:02.550 00 FEDC-I:STS FCP: Transfer Frame (#22, label=012) accepted by the spacecraft.
96-135-21:05:29.818 00 CMH-I:CMD IO(15): Received Command <C14013186ac00e001b019b008200004000000000000000000000000000000000000000000000> from VIRS:VIRS
96-135-21:05:30.659 00 FEDC-I:XFRM FCP: Spacecraft command - Frame #23
96-135-21:05:30.660 00 FEDC-I:CLTU FCP: EB90 006B042717C11812 6AC00E001BA4C14E A5D8A55AE55AA582 5AA55AA55AA55A38
96-135-21:05:33.671 00 FEDC-I:STS FCP: Transfer Frame (#23, label=013) accepted by the spacecraft.
96-135-21:06:08.860 00 FEDC-W:TLM Tue May 14 21:06:08 1996 FFP_dr11DM4; Q buffer wra
--%%-Emacs: *events-display* (Event Buffer Display--Forward)--L6411--Bot
009 COP1VER 96-135-21:00:27.506 VIRS:VIRS <C14009186ac009001b01Fc0082000000002000000000000000000000000000000000000000>
010 COP1VER 96-135-21:03:49.426 VIRS:VIRS <C14010186ac00b001b019e0082400000000000000000000000000000000000000000000000>
011 COP1VER 96-135-21:04:18.150 VIRS:VIRS <C14011186ac00c001b0159008280000000000000000000000000000000000000000000000>
012 COP1VER 96-135-21:04:58.796 VIRS:VIRS <C14012186ac00d001b0158008200008000000000000000000000000000000000000000000>
013 COP1VER 96-135-21:05:29.818 VIRS:VIRS <C14013186ac00e001b019b008200004000000000000000000000000000000000000000000>
--%%-Emacs: *command-status-display* (Command Status Display--Forward)--L127--Bot
STOL:asist
Procedure Input
ASIST_STARTUP/3855 : ;
ASIST_STARTUP/3856 : write "ASIST Version ",ASIST_VERSION,"
ASIST_STARTUP/3857 : return
ASIST_STARTUP/3858 :
ASIST_STARTUP/3859 : endproc
Operator Output
VC-1 DATA (VR-1 EVENT PLAYBACK TELEMETRY) IS DISABLED AT THIS W
Procedure USER_STARTUP completed.
ASIST Version 4.2.j is initialized as user asist
Procedure ASIST_STARTUP completed.
Echo is OFF.
Operator Input
acq i on
playback
FDS:
GMT: 96-135-22:20:54.000
PMT: 95-135-21:07:44.055
FDS Processor and Subsystem Status
Subsystem Cmd Counters
Page-FDS
Pri S/C
Pri ACS
Stby S/C
Stby ACS
Snap
Close
Packet Counters
Primary S/C is SIDE-A
Standby S/C is SIDE-A?
Primary ACS is SIDE-A
Standby ACS is SIDE-A?

- ASIST is good for you!!